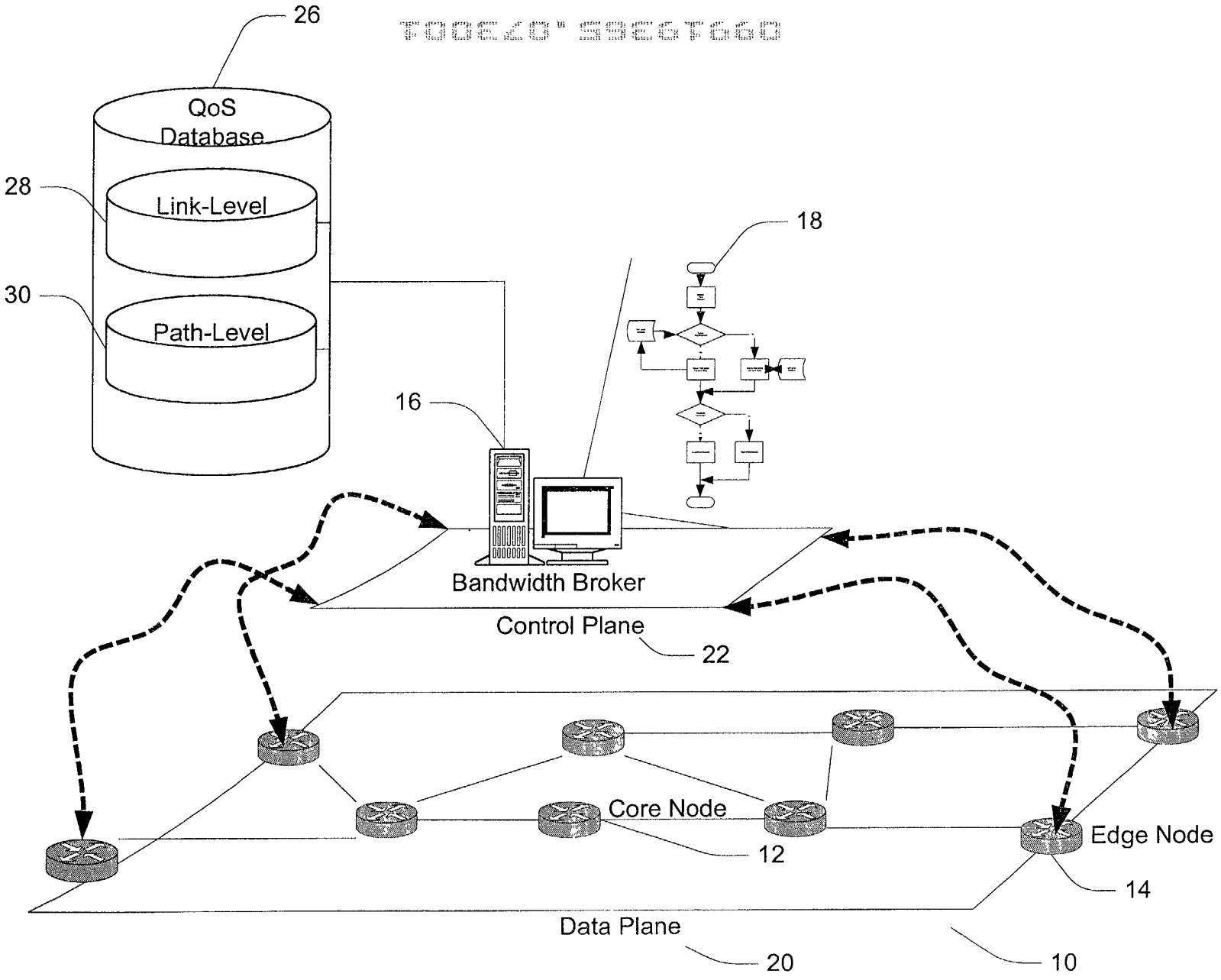
26
QoS Database
28
Link-Level
30
Path-Level
18
16
Bandwidth Broker
Control Plane
22
Core Node
12
Edge Node
14
Data Plane
20
10

FIG. 1

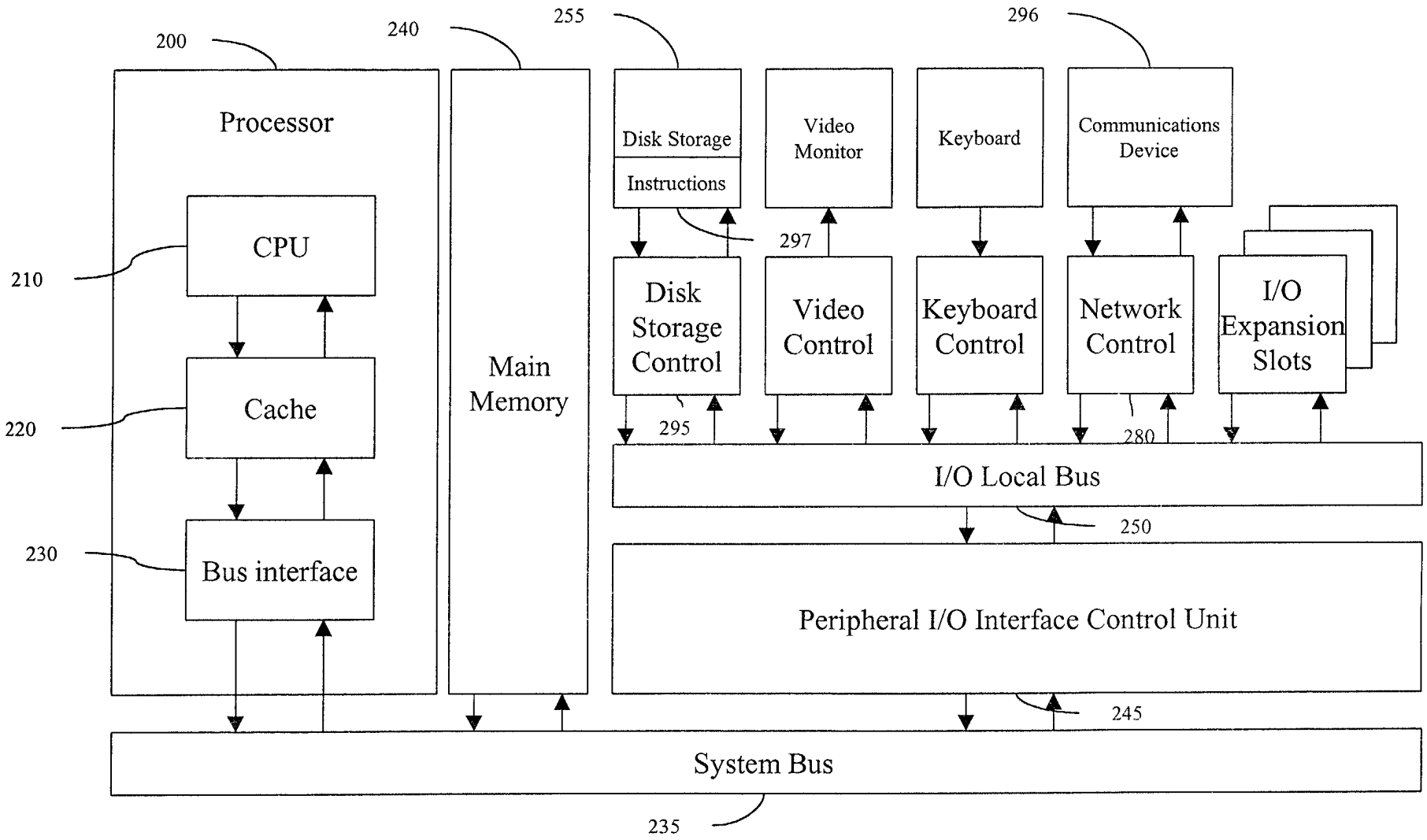
200
240
255
296
Processor
CPU
Cache
Bus interface
Main Memory
Disk Storage
Instructions
Video Monitor
Keyboard
Communications Device
297
Disk Storage Control
Video Control
Keyboard Control
Network Control
I/O Expansion Slots
210
220
230
295
280
I/O Local Bus
250
Peripheral I/O Interface Control Unit
245
System Bus
235

FIG. 2

0. Upon an arrival of a new flow f at a path p:
1. **case 1:** ($op_p == 0$ and $aqb_p \geq r_f$) ~300
2. $R_p \leftarrow R_p + r_f$; accept the flow; return. ~302
3. **case 2:** ($op_p == 0$ and $aqb_p < r_f$) ~304
4. request more quota on all the links l: $l \in p$ ~306
5. **case 3:** ($op_p > 0$) ~308
6. request bandwidth r_f on all critical links: $l \in cl_p$ ~310
7. **for** $l \notin cl_p$ ~312
8. if ($aqb_p < r_f$) request more quota ~314
9. **if** (all requests are granted) ~316
10. update Q_p if more quotas are allocated; ~318
11. $R_p \leftarrow R_p + r_f$; accept the flow; return. ~320
12. **else** reject the flow reservation set-up request. ~322

FIG. 3

0. Upon a path p requests r_p on a link l:
1. /* r_p can be a quota or a flow's request rate */
2. **case 1:** ($op_l == 0$ and $aq_l < r_p$) ~400 ~402
3. collect residual bandwidth: $rb_l \leftarrow C_l - \sum_{p:l \in p} R_p$;
4. **if** ($rb_l < r_p$) reject the request; return. ~404
5. **case 2:** ($op_l == 1$ and $rb_l < r_p$) reject the request; return.
6. /* The request can be honored */ ~406
7. **if** ($op_l == 0$ and $aq_l < r_p$) ~408
8. 410 ~ $op_l \leftarrow 1$; /* transition: normal $\rightarrow$ critical */
9. **for** ($p' : l \in p'$) ~412
10. $cl_{p'} \leftarrow cl_{p'} \cup l$; $op_{p'} \leftarrow op_{p'} + 1$; ~414
11. **case 1:** ($op_l == 0$) $aq_l \leftarrow aq_l - 1$ ~416
12. **case 2:** ($op_l == 1$) $rb_l \leftarrow rb_l - r_p$. ~418

FIG. 4

```
0.  Upon an existing flow f departs on a path p:
1.      R_p ← R_p − r_f; ~500
2.      if (op_p > 0) ~502
3.          for (l ∈ cl_p) ~504                          ~508
4.  506 ~       rb_l ← rb_l + r_f; recompute aq_l;
5.  510 ~       if (aq_l ≥ 0) /* transition: critical → normal */
6.                  for (p' : l ∈ p') ~512
7.                      op_p' ← op_p' − 1; set Q_p'; ~514
8.                      cl_p' ← cl_p' − l; ~516      ~518
9.      else if (op_p == 0 and p has excess quota)
10. 520 ~   Q_p ← Q_p − 1; /* return excess quota */
11.         for (l ∈ p) ~522
12.             aq_l ← aq_l + 1; ~524
```

FIG. 5

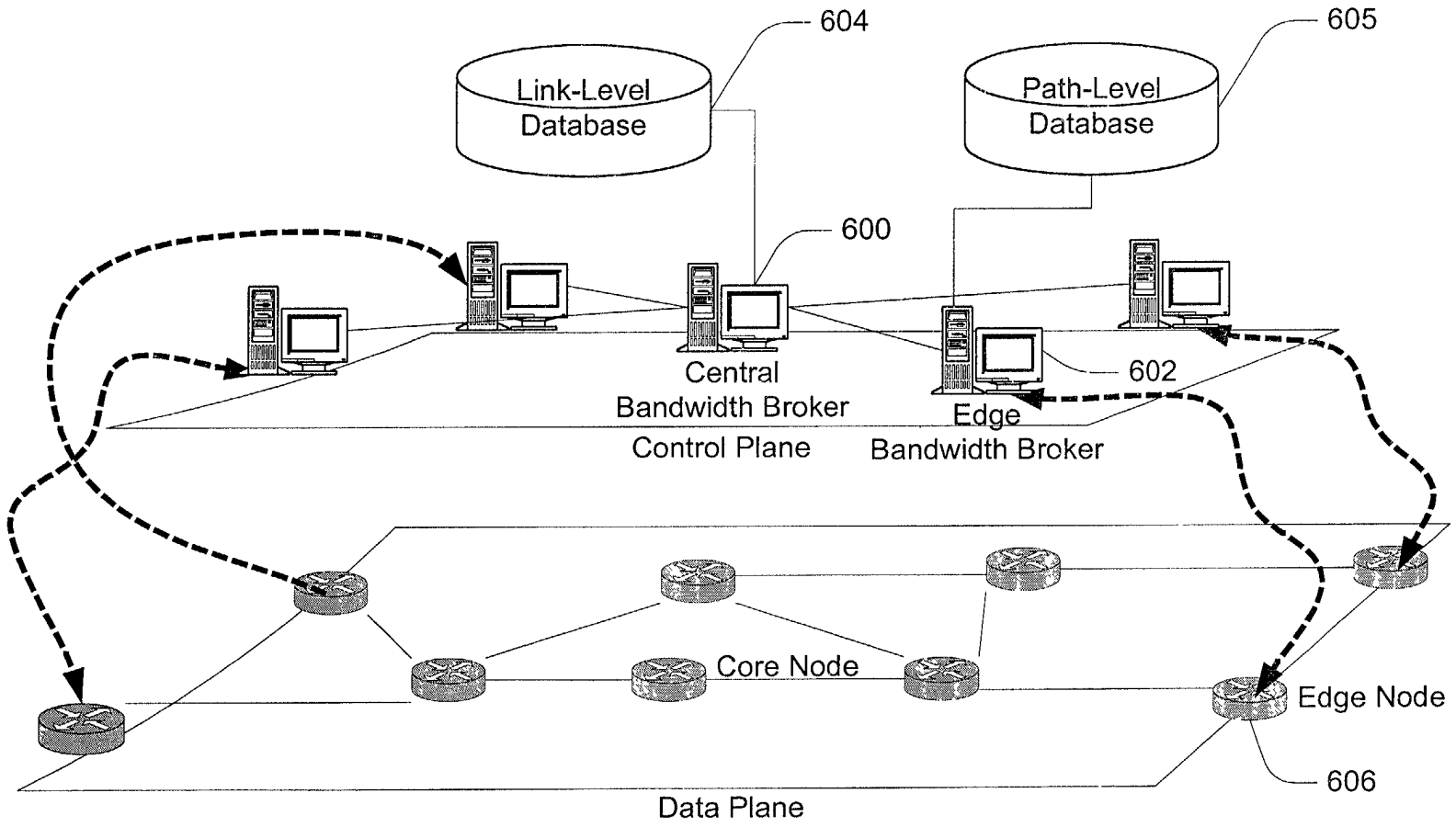

FIG. 6

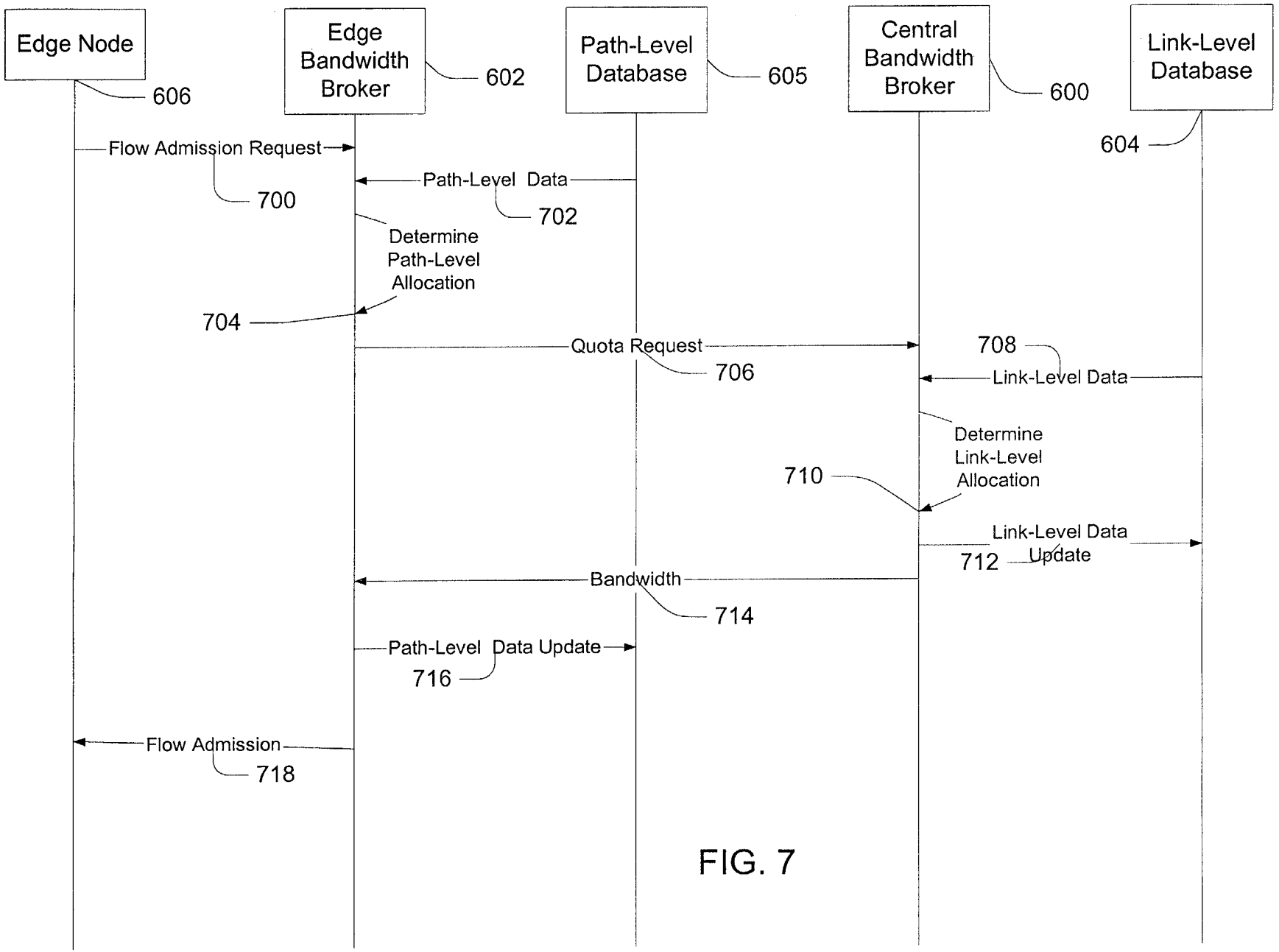
Edge Node
606
Edge Bandwidth Broker
602
Path-Level Database
605
Central Bandwidth Broker
600
Link-Level Database
604
Flow Admission Request
700
Path-Level Data
702
Determine Path-Level Allocation
704
Quota Request
706
708
Link-Level Data
Determine Link-Level Allocation
710
Link-Level Data Update
712
Bandwidth
714
Path-Level Data Update
716
Flow Admission
718

FIG. 7

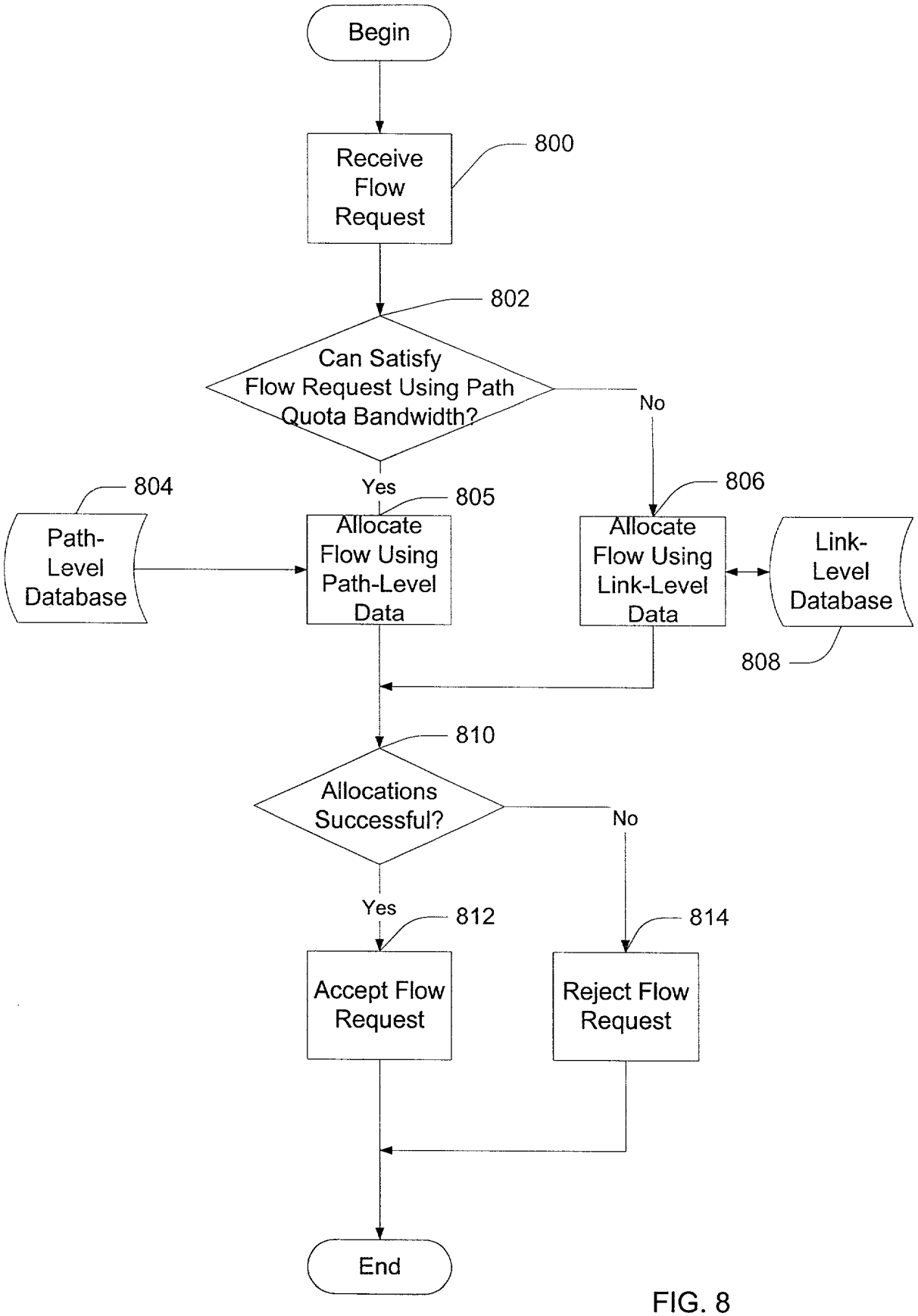
Begin
Receive Flow Request
800
802
Can Satisfy Flow Request Using Path Quota Bandwidth?
No
Yes
804
Path-Level Database
805
Allocate Flow Using Path-Level Data
806
Allocate Flow Using Link-Level Data
Link-Level Database
808
810
Allocations Successful?
No
Yes
812
Accept Flow Request
814
Reject Flow Request
End

FIG. 8